A feature-similarity model for product line engineering

Kaindl, Herman; Mannion, Mike

Published in:
Software Reuse for Dynamic Systems in the Cloud and Beyond

DOI:
10.1007/978-3-319-14130-5_3

Publication date:
2014

Document Version
Author accepted manuscript

A Feature-Similarity Model for Product Line Engineering

Hermann Kaindl[1] and Mike Mannion[2]

[1] Institute of Computer Technology, Vienna University of Technology, Vienna, Austria
kaindl@ict.tuwien.ac.at
[2] Executive Group, Glasgow Caledonian University
m.a.g.mannion@gcu.ac.uk

Abstract. Search, retrieval and comparison of products in a product line are common tasks during product line evolution. Feature modeling approaches do not easily support these tasks. This vision paper sets out a proposal for a *feature-similarity model* in which similarity metrics as used for example in *case-based reasoning* (CBR) are integrated with feature models. We describe potential applications for Product Line Scoping, Domain Engineering and Application Engineering.

Keywords: Product line engineering • feature-based representation • case-based reasoning • similarity metric • feature-similarity model.

1 Introduction

Software Product Line Engineering (SPLE) consists of: Product Line Scoping, Domain Engineering and Application Engineering (though in some frameworks [1] Product Line Scoping and Domain Engineering are both considered a part of a single core asset development activity). During these activities, we are interested in comparing the similarity of:

- a target product specification against existing product specifications;
- two or more existing product specifications against each other;
- how close an existing product matches the target market.

The information structure and content of feature models (regardless of notation) does not readily lend itself to compare similarity. Understanding what features are similar across different products, or what products are similar to other products, becomes very difficult as a product line evolves into tens, hundreds, maybe even thousands of products each with many features.

Case-based reasoning (CBR) is a development approach in which *software cases* are stored in a repository and new product development involves retrieving the most similar previous case(s) to the problem to be solved and then adapting it (them). Similarity matching, e.g. [2], is achieved by comparing some combination of *surface* features i.e. those provided as part of its description, *derived* features (obtained from a

product's description by inference based on domain knowledge) and *structural* features (represented by complex structures such as graphs or first-order terms). An overall similarity measure is computed from the weighted similarity measures of different elements. Efficient implementations for commonly used similarity metrics are readily available, so that the computational effort for search and retrieval of similar products has little impact on the efficiency of this approach. The key issue is the (manual) effort for adapting similar cases found and retrieved.

This vision paper sets out a proposal for a *feature-similarity model* in which similarity metrics as used in *case-based reasoning* (CBR), *information retrieval* or *service discovery* are integrated with feature models. We describe potential applications for Product Line Scoping, Domain Engineering and Application Engineering.

2 Related Work

Much work on feature models (see [3]) has focused on modelling representations and traversal algorithms that lend themselves to automation, and there has been little work on the relationships between product line feature models and similarity metrics.

The ReDSeeDS project (`http://www.redseeds.eu`) developed a specific similarity metric including textual, semantic and graph-based components [4]. Requirements *representations* are compared (e.g. requirements specifications or models) rather than requirements [5]. Reuse can be based on a *partial* requirements specification rather than a "complete" requirements specification [6]. The specification of new requirements can be facilitated by reusing related requirements from a retrieved software product, and the implementation information (models and code) of (one of) similar problems can be taken for reuse and adapted to the newly specified requirements. There are well-defined reuse processes, tightly connected with tool support (in parts) [7].

Similarity-based (*Web) service discovery* and (*Web) service retrieval* are also based on similarity metrics, see e.g. [8, 9]. (*Web) service matchmaking* uses additional heuristics [10], which may be adopted for whole software cases, and *semantic* service annotations and the use of ontologies allow for metrics apart from those based on text similarity [11]. Some approaches for an automated construction of feature models, based on similarity metrics, have been proposed [12, 13].

3 Contrasting SPLE and CBR

SPLE and CBR both support software product line engineering but address it differently. Table 1 summarizes these differences. In SPLE, the premise underlying feature model construction is that the rigour and consistency of the model structure can be used to derive new products from existing product elements. Model construction and maintenance costs (structure and content) are large but significantly reduce the costs of new product development and thus have large benefits for *reuse*. In CBR, the premise is that each product is constructed by retrieving and adapting similar cases already built. Product description construction costs are small whilst the cost of

adapting an existing case can vary. So, SPLE and CBR have differences in the costs of making software assets *reusable* and the benefits for *reusing* them. More details on this comparison can be found in [14].

Table 1. Costs vs. Benefits

	SPLE	**CBR**
Costs of Making *Reusable*	Substantial	Negligible
Benefits for *Reuse*	Facilitates automated product derivation	Facilitates finding similar cases for reuse

4 A Feature-Similarity Model

A *feature-similarity model* combines a feature model for managing and deriving new products and provides similarity values for several purposes. Figure 1 shows a small feature model for a mobile phone in which the *ability to make a call* and the *ability to display* are mandatory features. To make a call there are one or more alternatives and the display can be in black and white or in colour. Figure 1 also shows four products that contain different combinations of these features, and whilst the model shows only examples of functional features, it can be extended to include non-functional features.

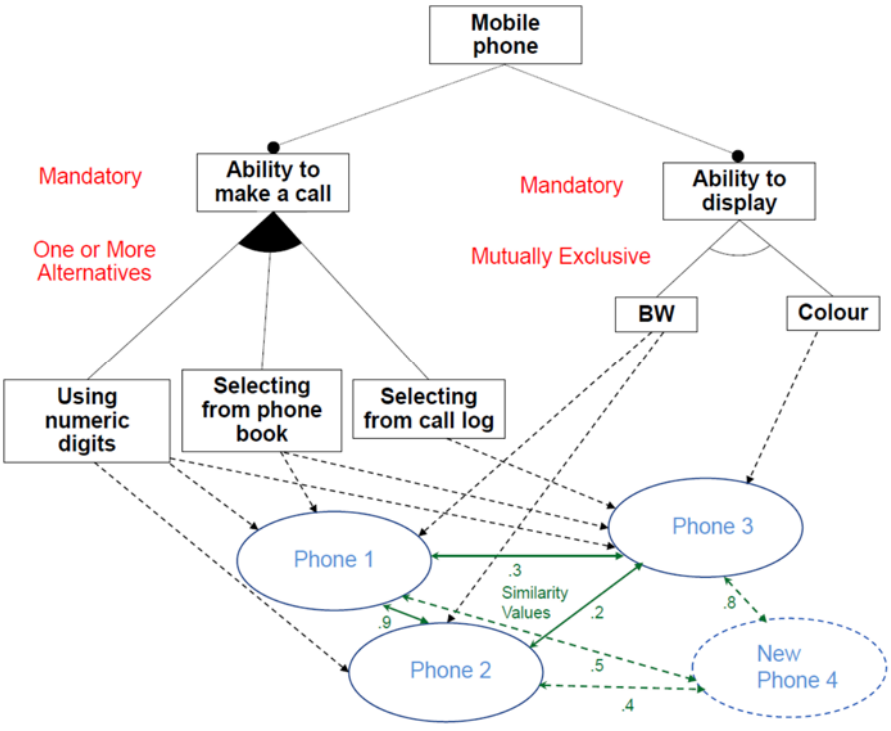

Figure 1: Integrated Model of a Mobile Phone Worked Example

The scores attached to the bidirectional links between the ellipses signify similarity scores between products e.g. Phone 1 has 2 features for making a call, using numeric digits or from a phone book, and a black and white display. Phone 2 has one way of making a call, using numeric digits, and a black and white (BW) display. Phone 3 has three ways of making a call and has a colour display. Intuitively Phone 1 and Phone 2 are more similar to each other than Phone 1 and Phone 3, or Phone 2 and Phone 3. This is played out in the similarity values Phone 1/Phone 2 (.9), Phone 1/Phone 3 (.3), Phone 2/Phone3 (.2). *New Phone 4* is not yet fully specified, but similarity scores compared to the other phones can be determined (cf. [5]) independently of whether this new product is derived from the given feature model or not.

The solid lines of the (standard) feature model are explicit, while the bidirectional links between the product ellipses are implicit i.e. they are computed on demand, not necessarily stored, especially not upfront. The dotted lines to the product ellipses should be stored, to enhance traceability during product derivation from the feature model thus clarifying which features are in which product.

A *feature-similarity model* (i) combines features and similarities (ii) provides direct similarity values between *features* (iii) facilitates comparing *feature combinations* in different products) (iv) facilitates comparing *entire product specifications*. The added value of an integrated model is the influence it will have during feature model construction on an engineer's deeper thinking about the extent to which one product is different to another, particularly when the overall similarity differences are small, why that is, and whether it is necessary or not. Each product line specification becomes a *searchable* asset and the similarity metrics enhance the search function.

The construction process for a feature-similarity model is (i) construct a feature model (ii) allocate features to products (iii) calculate similarity values between features (iv) calculate similarity values between specific feature combinations in different product specifications (v) calculate similarity values between entire product specifications. In practice these steps might be used iteratively to help with the construction of a product line to ensure that derivable products are sufficiently distinctive.

4.1 Product Line Scoping

Product Line Scoping is the process of identifying and bounding capabilities (products, features) and areas (subdomains, existing assets) of the product line where investment into reuse is economically useful and beneficial to product development [15]. The output is a scope definition, which can be seen as part of the specification the product line engineering project must eventually satisfy.

A feature-similarity model for product line scoping will focus on the features that represent the distinguishing characteristics of the product line, which are important for the market that is being targeted and which represent the product line boundary. That information will come from a variety of sources e.g. industry forecasts, sales and marketing intelligence, customer feedback, technology advances. Feature description detail and the complexity of the corresponding feature model structure will depend on the niche of the target market. For example in the fashion mobile phone market, the emphasis is on the shape, size and colour of the casing to reflect the values of a fash-

ion brand name, rather than the phone's functionality (often an existing model with minor modifications). By calculating the similarity value of the casing feature compared to a new target casing feature (step (iii) in the process above) we can inform the judgment about whether the product should be in the product line or not and how it might be best positioned in a market where look-and-feel govern distinctiveness.

When the number of target market characteristics and the number of products is small, product line scoping is tractable without much need for tool support. When the size or the level of complexity significantly increase, an automated similarity matching tool can become very valuable. Whilst scoping is normally an activity undertaken at the start of product line development it should continue as a product line evolves.

4.2 Domain Engineering

Domain Engineering encompasses domain requirements definition, domain analysis, domain design, domain implementation. Our focus here is domain requirements definition. Finding domain requirements can be difficult because existing requirements specifications (where they exist) can be written in different ways. A tool implementing a feature-similarity model (step (v)) can provide some support for identifying similarity between concepts across requirements specifications.

A feature-similarity model for Domain Engineering sets out all the features of the product line model. Adding new variability to an existing product line model can be difficult, and may not be required if there is already an existing alternative that seems close enough. Finding this alternative quickly in a large model can be very difficult and time-consuming. Making a judgment about the level of similarity of an existing alternative can help with making a commercial judgment on whether to proceed or not with including a new alternative. For example mobile phone product lines are becoming increasingly feature-rich and complex: the number of different ways of making a call is rapidly increasing beyond those set out in Figure 1. The ability to scan a number from a Quick Response code and the ability to scan a number from a jpeg image is similar. However if one of these alternatives is already available then a decision may be made not to include the other. Here we are calculating the similarity value of two alternative features (step (iii)), such as different ways of making a phone call. In practice it is highly unlikely that this approach would be adopted for all alternatives across all features, but rather, effort will be targeted on features that are valued by the customer but expensive to produce, and similar cheaper alternatives are available.

Feature-based SPLE most often lacks explicit representation of *non-functional* characteristics (e.g. performance, security, safety). In CBR, specific (text) searches based on similarity metrics can be an effective approach to uncovering such *cross-cutting concerns*.

4.3 Application Engineering

Application Engineering is the construction of a new product drawing on the assets that were developed during Domain Engineering. If a new product cannot be derived from a given product line model then often the model needs to be adapted so that

derivation can follow a defined process. Exceptionally, if the "new" features required are not going to be required in any other product, they can be added solely to the new product, and the product line model is not adapted. A feature-similarity model for Application Engineering sets out the features of the product line model that have been derived, through a selection process, for the new product.

In large-scale product lines a challenge is to know whether a new product being derived is similar to an existing one. The overall difference can be small but for a particular feature or feature combination it can be significant. Knowing the degrees of similarity can help with commercial judgments e.g. whether to introduce a new product into the market, use an existing product, or remove a product. Suppose there is a mobile phone product line targeted at teenagers, where the feature focus is the ability to make a call, the ability to display, the ability to take pictures and the ability to play music but the product line manager has decided to add a multi-person video-conferencing feature to the feature model. It will be sensible to know if this combination of features exists already in a different product line. By calculating the similarity value of the entire product specification (step (v)) against phones targeted at small-to-medium business enterprises (SMEs) that have all of these features, and by calculating the similarity values of a feature combination (step (iv)) of playing music and making multi-person video conference calls, we may discover that in the product line targeted at SMEs the phone we want already exists, albeit that the quality of the ability to play music is a little lower than desired but the quality of the multi-person video-conference facility is a little higher than desired.

In practice, there may not always be enough time for adapting a product line model so that the product derivation can take place, or there is insufficient time to complete the detailed selection process from the model. Then, the most similar products may be looked up to see whether the new product may be adapted directly based on them (so one attribute of a search tool for Application Engineering is to enable search by similarity threshold either for individual features and/or for entire products). In effect, this can lead to CBR reuse instead of working with feature models.

5 Discussion and Open-ended Questions

We do not prescribe here which set of specific similarity metrics to be used or how those metrics should be computed. Commonly-used general-purpose similarity metrics could be enhanced by other approaches e.g. it is possible to indicate whether a feature is a distinguishing characteristic of a product (e.g. display size) or not (e.g. number of default screen savers). We can distinguish between *alignable* differences and *non-alignable* differences [16] where alignable differences have the larger impact on people's judgments of similarity. Electronic Tablets and Mobile Phones have memory, processing power and a display (albeit of different sizes) which are recognized as *alignable* differences because they are characteristics defining computers. However, Mobile Phones provide the ability to make a telephone call without accessing the Internet, which Electronic Tablets usually do not, making this a non-alignable

difference. Placing a numerical value on the significance of alignable and non-alignable differences could be factored into the overall similarity metric.

Another approach is structural similarity i.e. a syntactic approach to matching normally based on the structure of a feature model. For example products having mutually exclusive features such as BW and colour display, respectively, e.g. *Phone 2* and *Phone 3*, may be considered less similar than products with the same feature, such as BW, e.g. *Phone 1* and *Phone 2*. Mutually exclusive features will make more of a difference in this regard than having one more feature of a kind where all the others are shared, e.g. *Phone 1* can make a call from a phone book, while *Phone 2* cannot.

We envisage that such similarity metrics of a feature-similarity model may also serve as objective functions for automated search in the space of systems defined by its feature model. Depending on what is to be optimised in terms of similarity, these may serve as cost functions or utility functions, respectively. Such approaches would fit into *search-based software engineering*, see [17].

A set of open ended questions includes:

- What are the thresholds for "similar" and for "different"?
- Which combination of similarity approaches might be suitable and when?
- What similarity metrics are worth computing and how should they be calculated ?
- How can similarity metrics be factored into existing process models for Product Line Scoping, Domain Engineering, and Application Engineering?

The deployment of similarity metrics for SPLE requires a degree of caution and prudence as with the use of any other software development metrics. Metrics provide a data reference point and will best serve managers and engineers when they are used in conjunction with data from other reference points. Be clear on what you are using the metric for, get general agreement in the organization on which metrics to use, and focus on only a few metrics – less is more.

6 Conclusion

A feature model does not facilitate search, retrieval or comparison of products in a product line, common tasks during product line evolution. To address this, we have set out ideas for enhancing feature models with similarity metrics in a new feature-similarity model. However we recognize that whatever metrics are used there will always be a need to map these numerical values on to an organisation's collective conceptual understanding of what similar and difference means in each context in which the metrics are being used.

7 References

1. A Framework for Software Product Line Practice, Version 5.0, http://www.sei.cmu.edu/productlines/frame_report (last accessed 6 Oct 2014)
2. Cover, T.M., Hart P.E.: Nearest Neighbour Pattern Classification. IEEE Trans. on Information Theory 13, 21–27 (1967)
3. Benavides, D., Felfernig, A, Galindo, J.A., Reinfrank, F.: Automated Analysis in Feature Modelling and Product Configuration. In: 13th International Conference on Software Reuse (ICSR '13), pp. 160–175 (2013)
4. Bildhauer, D., Horn, T., Ebert, J.: Similarity-driven software reuse. In: Proceedings of CVSM '09, pp. 31–36, IEEE (2010)
5. Kaindl, H., Svetinovic, D.: On confusion between requirements and their representations. Requirements Engineering 15, 307–311 (2010)
6. Kaindl, H., Smialek, M., Nowakowski, W.: Case-based Reuse with Partial Requirements Specifications. In: 18th IEEE International Requirements Engineering Conference (RE '10), pp. 399–400 (2010)
7. Kaindl, H., Falb, J., Melbinger, St., Bruckmayer, Th.: An Approach to Method-Tool Coupling for Software Development. In: Fifth International Conference on Software Engineering Advances (ICSEA '10), pp. 101–106, IEEE (2010)
8. Botelho, L., Fernández, A., Fires, B., Klusch, M., Pereira, L., Santos, T., Pais, P., Vasirani, M.: Service Discovery. In: Schumacher, M., Helin, H., Schuldt, H. (eds.) CASCOM: Intelligent Service Coordination in the Semantic Web, chapter 10, pp. 205–234. Birkhäuser, Basel (2008)
9. Czyszczon, A., Zgrzywa, A.: The MapReduce Approach to Web Service Retrieval. In: Computational Collective Intelligence: Technologies and Applications, LNCS, vol. 8083, pp. 517–526. Springer, Berlin Heidelberg (2013)
10. Klusch, M..: Semantic Web Service Coordination. In: Schumacher, M., Helin, H., Schuldt, H. (eds.) CASCOM: Intelligent Service Coordination in the Semantic Web, chapter 4, pp. 59–104. Birkhäuser, Basel (2008)
11. Becker, J., Oliver Müller, O., Woditsch, M.: An Ontology-Based Natural Language Service Discovery Engine – Design and Experimental Evaluation. In: 18th European Conference on Information Systems (ECIS '10) (2010)
12. Itzik, N., Reinhartz-Berger, I.: Generating Feature Models from Requirements: Structural vs. Functional Perspectives. In: REVE '14, SPLC Proceedings – Volume 2: Workshops, Demonstrations, and Tools, pp. 44–51 (2014)
13. Weston, N., Chitchyan, R., Rashid, A.: A framework for constructing semantically composable feature models from natural language requirements. In: 13th International Software Product Line Conference (SPLC '09), pp. 211–220 (2009)
14. Mannion, M., Kaindl, H.: Using Similarity Metrics for Mining Variability from Software Metrics. In REVE '14, SPLC Proceedings – Volume 2: Workshops, Demonstrations, and Tools, pp. 32–35 (2014)
15. John, I., Eisenbarth, M.: A Decade of Scoping: A Survey. In: 13th International Software Product Line Conference (SPLC '09), pp. 31–40 (2009)
16. McGill, A.L.: Alignable and nonalignable differences in causal explanations, Memory Cognition 30(3), 456–68 (2002)
17. Harman, M., Jia, Y., Krinke, J., Langdon, W., Petke, J., Zhang, Y.: Keynote: Search based software engineering for software product line engineering: a survey and directions for future work, In: 18th International Software Product Line Conference (SPLC '14), pp. 5–18 (2014)